Inductive Proof of Running Time for Binary Search

The goal is to prove that the running time of the binary search algorithm is $O(\log n)$.

Base Case:

Consider the smallest possible input, where the range $[lo, hi)$ contains only one element. In this case, the binary search algorithm will make exactly one comparison to determine the element, and the running time is $O(1)$, which is consistent with $O(\log n)$ for $n = 1$.

Inductive Step:

Assume that for a range of size n, the running time of the binary search algorithm is $O(\log n)$.

Now, consider a range of size $2n$. The algorithm works as follows:

1. Calculate the midpoint mid of the range.
2. Make one comparison to determine whether the search should continue in the left half or the right half of the range.
3. Recursively apply the binary search to the selected half, which has a size of n.

By the inductive hypothesis, the running time for a range of size n is $O(\log n)$. Therefore, for a range of size $2n$, the running time can be expressed as: $T(2n) = T(n) + O(1)$

Where $T(n)$ is the running time for the smaller range of size n, and $O(1)$ represents the time taken for the comparison and midpoint calculation.

Using the inductive hypothesis: $T(n) = O(\log n)$

So, $T(2n) = O(\log n) + O(1) = O(\log n)$

Since $2n$ is simply another way of expressing the size of the input in terms of n, we can generalize this to: $T(n) = O(\log n)$

Conclusion:

By induction, we have shown that if the running time of the binary search algorithm is $O(\log n)$ for a range of size n, then it is also $O(\log n)$ for a range of size $2n$. Thus, the running time of the binary search algorithm is $O(\log n)$ for all n.

This completes the inductive proof that the running time of the binary search algorithm is $O(\log n)$.